What is claimed is:

1: A method of managing a lock utilized by a thread comprising:

selecting an action to perform upon the lock, wherein the action is selected from a group comprising:

acquiring the lock,

trying to acquire the lock, and

releasing the lock;

asynchronously querying the current state of a lock, having a multi-value state;

speculatively determining the next state of the lock; and

attempting to transition the lock from the queried current state to the speculatively determined next state.

2: The method of claim 1, further including, if the state transition fails and if the selected action was either acquiring or releasing the lock, repeating, until the state transition succeeds:

asynchronously querying the current state of the lock;

speculatively determining the next state of the lock;

attempting to transition the lock from the queried current state to the speculatively determined next state.

3: The method of claim 2, further including, if

the state transition succeeds,

the selected action is acquiring the lock, and

the speculatively determined next state represents the acquisition of the lock, indicating the acquisition of the lock.

4: The method of claim 3, further including, if

the state transition succeeds,

the selected action is acquiring the lock, and

the speculatively determined next state does not represent the acquisition of the lock,

adding the thread to the end of a queue of threads waiting to acquire the lock;

waiting to receive notification that the thread may acquire the lock; and

indicating the acquisition of the lock.

5: The method of claim 2, further including, if

the state transition succeeds, and

the selected action is releasing the lock,

determining the number of threads in a queue to acquire the lock utilizing the speculatively determined next state of the lock.

6: The method of claim 5, further including, if the queue includes at least a first thread,
removing the first thread from the queue; and
notifying the first thread that the first thread has acquired the lock.

7: The method of claim 1, further including, if
the selected action is trying to acquire the lock and
the state transition fails,
indicating that the lock was unable to be acquired.

8: The method of claim 1, further including, if
the state transition succeeds and
the selected action is trying to acquire the lock,
indicating the acquisition of the lock.

9: The method of claim 1, further including, if
the state transition succeeds,
the selected action is acquiring the lock, and
the speculatively determined next state represents the acquisition of the lock,
indicating the acquisition of the lock.

10: The method of claim 9, further including, if

the state transition succeeds,

the selected action is acquiring the lock, and

the speculatively determined next state does not represent the acquisition of the lock,

adding the thread to the end of a queue of threads waiting to acquire the lock;

waiting to receive notification that the thread may acquire the lock; and

indicating the acquisition of the lock.

11: The method of claim 1, further including, if

the state transition succeeds, and

the selected action is releasing the lock,

determining the number of threads in a queue to acquire the lock utilizing the speculatively determined next state of the lock.

12: The method of claim 11, further including, if the queue includes at least a first thread,

removing the first thread from the queue; and

notifying the first thread that the first thread has acquired the lock.

13: The method of claim 1, wherein the thread includes:

a unique thread identifier;

a next thread field to facilitate access to the next thread in a queue of threads waiting to acquire the lock; and

the thread is only capable of waiting for a single lock at a time.

14: The method of claim 1, wherein the action of acquiring the lock includes the inability to timeout or fail to acquire the lock.

15: The method of claim 1, wherein the lock's current state may change between

asynchronously querying the current state of the lock; and

attempting to transition the lock from the queried current state to the speculatively determined next state.

16: An apparatus comprising:

a lock, having a multi-state value, including:

a flag value, a first thread value, and a last thread value; and

a lock acquirer, which is capable of performing an acquisition of the lock via

asynchronously querying the current state of the lock;

speculatively determining the next state of the lock; and

attempting to transition the lock from the queried current state to the speculatively determined next state.

17: The apparatus of claim 16, wherein the lock acquirer is further capable of performing two general actions, including acquiring the lock, trying to acquire the lock; and

wherein, if the state transition fails and the general action is acquiring the lock, the lock acquirer is further capable of, repeating, until the state transaction succeeds:

asynchronously querying the current state of the lock;

speculatively determining the next state of the lock; and

attempting to transition the lock from the queried current state to the speculatively determined next state.

18: The apparatus of claim 17, wherein, if the state transition fails and the general action is trying to acquire the lock, the lock acquirer is further capable of,

indicating that the lock was unable to be acquired.

19: The apparatus of claim 18, wherein, if the state transition succeeds and the general action is trying to acquire the lock, the lock acquirer is further capable of,

indicating that the lock was acquired.

20: The apparatus of claim 16, wherein, if

the state transition succeeds,

the general action is acquire the lock, and

the speculatively determined next state represents the acquisition of the lock, the lock acquirer is further capable of,

indicating that the lock was acquired.

21: The apparatus of claim 20, wherein, if

the state transition fails,

the general action is acquire the lock, and

the speculatively determined next state does not represent the acquisition of the lock,

the lock acquirer is further capable of,

adding the thread to the end of a queue of threads waiting to acquire the lock;

waiting to receive notification that the thread may acquire the lock; and

indicating the acquisition of the lock.

22: The apparatus of claim 21, wherein the lock acquirer is unable to timeout of fail if the selected general action is acquiring the lock.

23: An apparatus comprising:

a lock, having a multi-state value, including:

a flag value, a first thread value, and a last thread value; and

a lock releaser, which is capable of releasing a hold on the lock via

asynchronously querying the current state of the lock;

speculatively determining the next state of the lock; and

attempting to transition the lock from the queried current state to the speculatively determined next state.

24: The apparatus of claim 23, wherein, if the state transition fails, the lock releaser is further capable of, repeating, until the state transaction succeeds:

asynchronously querying the current state of the lock;

speculatively determining the next state of the lock; and

attempting to transition the lock from the queried current state to the speculatively determined next state.

25: The apparatus of claim 23, wherein, if the state transition succeeds, the lock releaser is further capable of determining the number of threads in a queue of threads waiting to acquire the lock utilizing the speculatively determined next state of the lock.

26: The apparatus of claim 25, wherein, if the queue includes at least a first thread, the lock releaser is further capable of:

removing the first thread from the queue; and

notifying the first thread that the first thread has acquired the lock.

27: The apparatus of claim 26, wherein the lock releaser is capable of removing the first thread from the queue utilizing a thread having:

a unique thread identifier; and

a next thread value to facilitate access to the next thread in the queue.

28: The apparatus of claim 23, wherein the lock is capable of changing state in between the time the lock releaser

asynchronously queries the current state of the lock; and

attempts to transition the lock from the queried current state to the speculatively determined next state.

29: An article comprising:

a storage medium having a plurality of machine accessible instructions, wherein when the instructions are executed, the instructions provide for:

selecting an action to perform upon a lock utilized by a thread, wherein the action is selected from a group comprising:

acquiring the lock,

trying to acquire the lock, and

releasing the lock;

asynchronously querying the current state of a lock, having a multi-value state;

speculatively determining the next state of the lock; and

attempting to transition the lock from the queried current state to the speculatively determined next state.

30: The article of claim 29, further including instructions providing for, if the state transition fails and if the selected action was either acquiring or releasing the lock, repeating, until the state transition succeeds:

asynchronously querying the current state of the lock;

speculatively determining the next state of the lock;

attempting to transition the lock from the queried current state to the speculatively determined next state.

31: The article of claim 30, further including instructions providing for, if

the state transition succeeds,

the selected action is acquiring the lock, and

the speculatively determined next state represents the acquisition of the lock, indicating the acquisition of the lock.

32: The article of claim 31, further including instructions providing for, if

the state transition succeeds,

the selected action is acquiring the lock, and

the speculatively determined next state does not represent the acquisition of the lock,

adding the thread to the end of a queue of threads waiting to acquire the lock;

waiting to receive notification that the thread may acquire the lock; and

indicating the acquisition of the lock.

33: The article of claim 30, further including instructions providing for, if

the state transition succeeds, and

the selected action is releasing the lock,

determining the number of threads in a queue to acquire the lock utilizing the speculatively determined next state of the lock.

34: The article of claim 33, further including instructions providing for, if the queue includes at least a first thread,

removing the first thread from the queue; and

notifying the first thread that the first thread has acquired the lock.

35: The article of claim 29, further including instructions providing for, if
the selected action is trying to acquire the lock and
the state transition fails,
indicating that the lock was unable to be acquired.

36: The article of claim 29, further including instructions providing for, if
the state transition succeeds and
the selected action is trying to acquire the lock,
indicating the acquisition of the lock.

37: The article of claim 29, further including instructions providing for, if
the state transition succeeds,
the selected action is acquiring the lock, and
the speculatively determined next state represents the acquisition of the lock,
indicating the acquisition of the lock.

38: The article of claim 37, further including instructions providing for, if
the state transition succeeds,
the selected action is acquiring the lock, and
the speculatively determined next state does not represent the acquisition of the
lock,

adding the thread to the end of a queue of threads waiting to acquire the lock;
waiting to receive notification that the thread may acquire the lock; and
indicating the acquisition of the lock.

39: The article of claim 29, further including instructions providing for, if
the state transition succeeds, and
the selected action is releasing the lock,
determining the number of threads in a queue to acquire the lock utilizing the speculatively determined next state of the lock.

40: The article of claim 39, further including instructions providing for, if the queue includes at least a first thread,
removing the first thread from the queue; and
notifying the first thread that the first thread has acquired the lock.

41: The article of claim 29, wherein the thread includes:
a unique thread identifier;
a next thread field to facilitate access to the next thread in a queue of threads waiting to acquire the lock; and
the thread is only capable of waiting for a single lock at a time.

42: The article of claim 29, wherein the action of acquiring the lock includes the inability to timeout or fail to acquire the lock.

43: The article of claim 29, wherein the lock's current state may change between

asynchronously querying the current state of the lock; and

attempting to transition the lock from the queried current state to the speculatively determined next state.

44: A system comprising:

a memory element, capable of storing a queue of threads, each thread including

a unique thread identifier, and a next thread value to facilitate access to the next thread in the queue;

a lock, having a multi-state value, including:

a flag value, a first thread value, and a last thread value; and

a lock acquirer, which is capable of performing an acquisition of the lock via

asynchronously querying the current state of the lock;

speculatively determining the next state of the lock; and

attempting to transition the lock from the queried current state to the speculatively determined next state.

45: The system of claim 44, wherein the lock acquirer is further capable of performing two general actions, including acquiring the lock, trying to acquire the lock; and

wherein, if the state transition fails and the general action is acquiring the lock, the lock acquirer is further capable of, repeating, until the state transaction succeeds:

asynchronously querying the current state of the lock;

speculatively determining the next state of the lock; and

attempting to transition the lock from the queried current state to the speculatively determined next state.

46: The system of claim 45, wherein, if the state transition fails and the general action is trying to acquire the lock, the lock acquirer is further capable of,

indicating that the lock was unable to be acquired.

47: The system of claim 46, wherein, if the state transition succeeds and the general action is trying to acquire the lock, the lock acquirer is further capable of,

indicating that the lock was acquired.

48: The system of claim 44, wherein, if

the state transition succeeds,

the general action is acquire the lock, and

the speculatively determined next state represents the acquisition of the lock,

the lock acquirer is further capable of,

indicating that the lock was acquired.

49: The system of claim 48, wherein, if

the state transition fails,

the general action is acquire the lock, and

the speculatively determined next state does not represent the acquisition of the lock,

the lock acquirer is further capable of,

adding the thread to the end of the queue of threads waiting to acquire the lock;

waiting to receive notification that the thread may acquire the lock; and

indicating the acquisition of the lock.

50: The system of claim 49, wherein the lock acquirer is unable to timeout of fail if the selected general action is acquiring the lock.

51: A system comprising:

a memory element, capable of storing a queue of threads, each thread including

a unique thread identifier, and a next thread value to facilitate access to the next thread in the queue;

a lock, having a multi-state value, including:

a flag value, a first thread value, and a last thread value; and

a lock releaser, which is capable of releasing a hold on the lock via

asynchronously querying the current state of the lock;

speculatively determining the next state of the lock; and

attempting to transition the lock from the queried current state to the speculatively determined next state.

52: The system of claim 51, wherein, if the state transition fails, the lock releaser is further capable of, repeating, until the state transaction succeeds:

asynchronously querying the current state of the lock;

speculatively determining the next state of the lock; and

attempting to transition the lock from the queried current state to the speculatively determined next state.

53: The system of claim 51, wherein, if the state transition succeeds, the lock releaser is further capable of determining the number of threads in the queue of threads waiting to acquire the lock utilizing the speculatively determined next state of the lock.

54: The system of claim 53, wherein, if the queue includes at least a first thread, the lock releaser is further capable of:

removing the first thread from the queue; and

notifying the first thread that the first thread has acquired the lock.

55: The system of claim 51, wherein the lock is capable of changing state in between the time the lock releaser

asynchronously queries the current state of the lock; and

attempts to transition the lock from the queried current state to the speculatively determined next state.